Digital Twin-Empowered Resource Allocation for 6G-Enabled Massive IoT

Elif Bozkaya*, Berk Canberk† and Stefan Schmid‡
*Department of Computer Engineering, National Defence University Naval Academy, Istanbul, Turkey
†School of Computing, Engineering and The Built Environment, Edinburgh Napier University, United Kingdom
‡Department of Electrical Engineering and Computer Science, Technical University of Berlin, Berlin, Germany
Email: ebozkaya@dho.edu.tr, b.canberk@napier.ac.uk, stefan.schmid@tu-berlin.de

2023 IEEE International Conference on Communications Workshops (ICC Workshops) | 979-8-3503-3307-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCWORKSHOPS57953.2023.10283649

***Abstract*—6G technology is expected to lead to an unpredictable increase of Internet of Things (IoT) devices. The need for maintaining continuous connectivity of these devices has in turn led to re-thinking of the traditional design of wireless networks. In particular, the integration of 6G and Digital Twin (DT) is expected to reshape the network management as it offers powerful features in design, development and optimization processes. DT is a digital representation of physical entities, which are designed around a two-way information flow. Therefore, this technology not only collects data, and employs intelligent learning methods by performing complex computations, but it also can send feedback to improve system performance for 6G-enabled massive IoT. However, deploying such a technology requires addressing complex challenges such as limited resources, seamless connectivity and lack of trust between end users and network edge. To address these challenges, we formulate the resource allocation problem including edge computation and service migration in 6G-enabled massive IoT. The contributions are threefold: First, our DT-empowered architecture is proposed that uses the real-time and historical data from end users to find the best allocation at a user. Second, it studies the impact of trust relationship between computing entities to prevent the unauthorized accesses and provides an authentication procedure. Third, it describes a Multi-Agent Reinforcement Learning (MARL) algorithm that consists of cooperative agents and aims to find the best resource allocation strategy by minimizing task processing latency. We validate the proposed DT-empowered architecture to show the reduced processing latency compared to traditional benchmark methods.**

I. Introduction

With recent advancements in wireless technologies, the Internet of Things (IoT) paradigm is enabling exciting applications such as smart cities, autonomous cars, smart grids, augmented/virtual reality, which are massive in number and computation, and latency sensitive. Due to the strict requirements of such applications, there is a need for technological advancements in next generation wireless communication toward 6G [1]. These technologies need to support massive number of devices for seamlessly information exchange with high data rates and low latency.

One of the most challenging issues is to support connectivity and Quality of Service (QoS) requirements for data and computation intensive and latency sensitive systems at large scale. Meanwhile, processing massive data and analyzing them for accurate and right decision cause major challenges at the devices with limited computing capability. Thus, establishing such a 6G-enabled massive IoT system for efficient resource utilization and optimization is accelerating the way toward digital transformation.

In this context, Digital Twin (DT) is one of the key enabling technologies. Recently, DT has been attracting extensive attention in both academia and industry. It provides a digital world consisting of the replicas of physical objects with a real-time two way interaction. DT completely differs from the traditional methods since DT offers a real time feedback, predictions about the system in the future and allows to run several simulations [2]. To improve system performance, DT can use both real-time and historical data for advanced analytics, this increases the accuracy of the intelligent learning model, monitors the system in real-time and offers better understanding for the user behavior pattern throughout the network's lifetime.

Motivated by these trends, we introduce a DT-empowered resource allocation framework that aims to utilize the resources at the edge of the networks so that end users can offload computation intensive tasks to the nearby edge servers for low latency and seamless connectivity. We propose a Multi-Agent Reinforcement Algorithm (MARL), which is based on coordination among agents and learns the optimal resource allocation strategy to maximize their joint utility. To achieve this, we determine stochastic games that model dynamic interactions where the environment changes according to the choices of users. We also investigate the service migration problem in order to reduce the possibility of service interruptions and avoid large number of authentication procedure.

Succinctly, the main contributions of this paper are as follows.

- We construct a DT-empowered resource allocation framework for efficient task offloading and service migration in 6G-enabled massive IoT networks.
- We devise a Multi-Agent Reinforcement Learning (MARL) algorithm, taking advantage of cooperation among end users to minimize task processing latency. We also propose a trust relationship between computing entities to prevent the unauthorized accesses.
- We carry out extensive experiments to evaluate the performance of the proposed DT-empowered resource allocation framework based on MARL. We compare the task processing latency in terms of joint learning and

independent learning strategies and also show significant gains over traditional benchmark methods.

The rest of this paper is organized as follows. Section II introduces the previous works in 6G-enabled massive IoT. Section III presents the overview of our DT-empowered system model while Section IV discuss the proposed MARL algorithm. Further, in Section V, we evaluate the results and in Section VI, we conclude the paper.

II. Related Work

We classify the previous works into two categories: (i) edge computing perspective with resource allocation and (ii) DT approaches and solutions.

Liu et al. propose an offloading strategy for an optimal resource allocation. Each user is designed as a learning agent to find an optimal offloading decision on local computing or edge computing. To achieve this, multi-agent Q-learning algorithm is developed based on independent learners [3]. Kiran et al. investigate dynamic task offloading and resource allocation problem to minimize the delay and energy. The authors integrate the software defined network (SDN) with edge computing services and use Q-learning and cooperative Q-learning based reinforcement learning algorithm implemented by a centralized SDN controller [4]. Ahmed et al. present a Stackelberg Game based dynamic resource sharing model between MEC servers and edge server providers to maximize the usage of underutilized edge resources so that the model balances the workload among servers [5]. Similarly, Xiong et al. propose a resource allocation strategy for the IoT edge computing system to minimize the task completion task and improve the efficiency of resource utilization. The authors solve the resource allocation problem by implementing a deep reinforcement learning approach [6].

There are additional studies on resource allocation in traditional solutions in 6G-enabled massive IoT. For example, Zhang et al. investigate the task offloading, resource allocation and service caching problems in DT-enabled edge network and formulates the problem as a mixed-integer programming to minimize the energy consumption of the overall system [7]. Van Huynh et al. use the DT models for the computation capacity of servers and optimize the resource allocation. It is aimed to minimize end-to-end latency by formulating a mixed integer optimization problem and, then solving it in an iterative manner [8]. Guo et al. propose a DTEN architecture for privacy awareness and resource allocation with a federated reinforcement learning algorithm. The authors use DT models for monitoring and optimization in 6G Industrial Internet of Things (IIoT) network [9]. Aside from the traditional context, we focus on DT solutions to deal with resource allocation problems with service migration management since recent studies are working on general task offloading in edge computing and the DT technology is still not adopted. We propose a fully DT-empowered resource allocation strategy and implement an authentication mechanism to prevent unauthorized access.

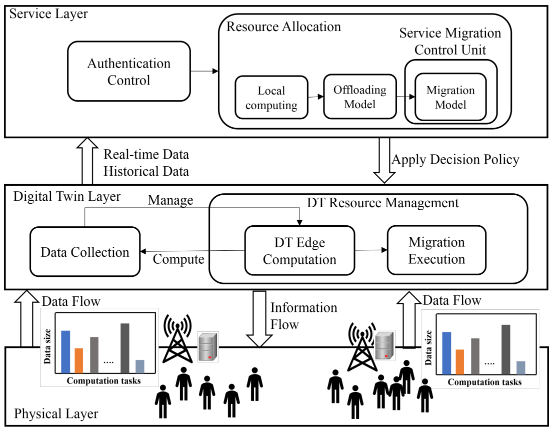

Fig. 1. Overview of the DT-empowered system model.

III. System Model

In this section, we first present our intelligent DT-empowered system model as seen in Fig. 1. Our system model is composed of three layers: (i) physical layer, (ii) digital twin layer and (iii) service layer.

A. Physical Layer

In this paper, we consider a layered-based architecture containing multiple small cell base stations (SBS) with edge server capability and multiple end users (i.e, smartphone, sensors) in a given region. End users, indicated as $M = \{1, 2, .., i, .., m\}$, request a broad range of computation tasks such as video streaming, augmented reality etc. Each task is composed of four elements $\Omega_i = \{C_i, B_i, \theta_i, Z_i\}$, where C_i represents the number of CPU cycles required by the task, B_i is the data size, $\theta_i \in [0, 1]$ is a variable which denotes the offloading decision and Z_i is the acceptable latency to achieve the task.

B. Digital Twin Layer

In our DT-empowered model, digital copies are created for each physical entities. The DT layer is responsible to interact with service layer on behalf of physical entities. Both edge computing and service migration management require to receive the location information of end users so that this information is retrieved from the DT instead of the physical entities, making the network more adaptable for dynamic computing [10].

In this paper, the motivations of using DT is as follows:

- An intelligent learning model is applied in DT so that both historical data and real-time data are used to predict the future requests and improve the system performance throughout the network's lifetime.
- Extracting the user features/behavior patterns in DT helps better characterization of physical entities and enable more efficient resource management.
- When multiple end users request the same computation tasks, the required information is received from the DT only once, which tremendously reduces the network traffic load.

In the next subsections, functionalities of the service layer will be detailed.

C. Authentication Control

One of the critical steps in this system is to process massive amount of heterogeneous data generated by multiple users. While DT-empowered and edge computing-based infrastructure provides low latency and optimal resource utilization with high computing and storage capabilities, the solutions also need to establish a trust relationship between computing entities to prevent unauthorized access. Thus, we implement an authentication mechanism before resource allocation. Our DT model works by authenticating clients and servers using digital certificates and unique keys. The model can be also extended for data integrity and privacy. Accordingly, the proposed approach establishes a trust relationship with computing entities. The users and edge servers authenticate using certificates distributed by a trusted authority (TA). After the authentication, the users can offload the computing tasks to the network edge.

Fig. 2 gives the steps of authentication procedure. (1) Each entity requests a certificate from the TA. (2) TA is mainly responsible for distributing a certificate including, expiration time, the user ID and public key of the user. The unique certificate is signed/encrypted by the TA with its private key by using public key encryption. In the figure, the certificate is denoted by $C_1 = E(K_{TA}^-(Time1, ID_1, K_1^+))$, where K_{TA}^- and K_1^+ are the private key of the TA and public key of the user ID_1, respectively, and $Time1$ is the expiration time. (3-4) The user and edge node send their certificates to each other. (5) Then, the user signs/encrypts $Hello$ message and a random number, $R1$ with its private key, K_1^-. (6) The edge node decrypts the message with the user public key, K_1^+. (7) The edge node sends $(Hello - R1)$ message and random number $R2$ in response to the user message by encrypting with its private key, K_e^-. (8) Similarly, the user decrypts the message with the edge node public key, K_e^+. (9) At the last step, the user signs/encrypts $R2$ with its private key, K_1^- and (10) the edge node decrypts the message with the user public key, K_1^+. Thus, the edge node and user authenticate each other.

D. Resource Allocation

1) Local Computing: The computation tasks can be executed locally or partially by the end user so that the amount of computing task θ_i performed by end user i is $B_i \times \theta_i$, where B_i is the task size. Thus, the local computing time of the end user i can be expressed as [11].

$$T_i^{local} = \frac{C_i B_i \theta_i}{f_i^{local}} \tag{1}$$

where f_i^{local} is the CPU frequency of end user i and C_i is the number of CPU cycles of the computing task. If the entire computing task is executed locally, then $\theta_i = 1$ and the processing latency is equal to the local computing time, $T_i^{proc} = T_i^{local}$.

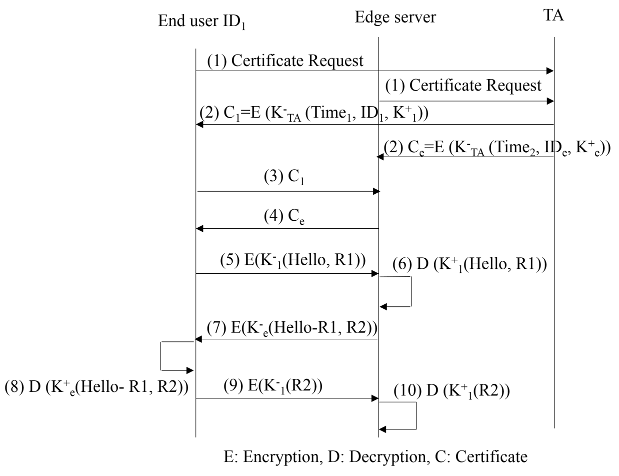

Fig. 2. Authentication procedure between the user and edge server.

2) Offloading Model: When DT of the end user decides to offload the task to the edge server, the computing task is transferred to the edge server incurring uplink communication latency. The transmission latency from end user to the edge server is given as follows:

$$T_i^{trans} = \frac{B_i(1-\theta_i)}{R_i} \tag{2}$$

where R_i the maximum achievable uplink transmission rate of end user i. According to the Shannon's theorem, R_i can be expressed as follows.

$$R_i = W \log_2(1 + SINR) \tag{3}$$

where W is the channel bandwidth and SINR represents the Signal to Interference plus Noise Ratio between a user and BS which can be calculated by:

$$SINR = \frac{P_i d_{j,i}^{-\alpha}}{\sigma^2 + \sum_{k \in \Phi \setminus \{i\}} P_k d_{j,k}^{-\alpha}} \tag{4}$$

where σ^2 noise power, Φ is the set of interferer users k to the BS j, P_i is the transmission power of end user i, $d_{j,i}$ is the distance between user and BS, α is the path loss exponent.

Then, execution time of the computing task is calculated as follows:

$$T_i^{exe} = \frac{C_i B_i (1-\theta_i)}{f_i^{edge_j}} \tag{5}$$

where $f_i^{edge_j}$ is the computing resources allocated by edge server j to the end user i.

As a result, processing latency is equal to the sum of transmission latency and execution time of the task.

$$T_i^{proc} = T_i^{trans} + T_i^{exe} \tag{6}$$

3) Migration Model: In order to guarantee service continuity, our framework includes *service migration control unit* since the computing tasks might need a large number of migrations due to the mobility of users and limited coverage area of BSs. This also results in a large number of

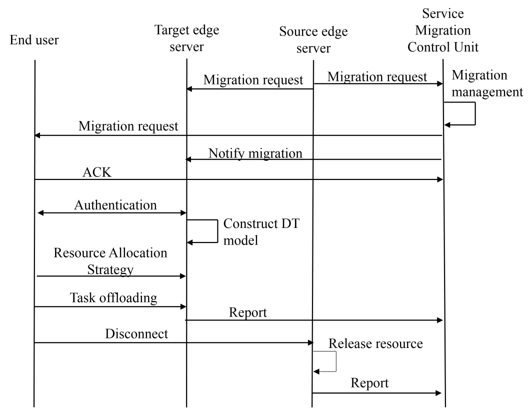

Fig. 3. Sequence of DT-based service migration procedure.

authentication procedure executions. The control unit has all network information and serves as an orchestrator. As an end user approaches to the edge of a cell, the Radio Network Interface Service (RNIS) running on the edge server informs the target edge server that the user is about to perform the service migration [12]. According to the user location and, required computation, storage, and network resources, service migration control unit decides whether to migrate the service. In Fig. 3, we illustrate the service migration procedure between end user and edge servers.

When the task is decided to migrate to the target edge server, the remaining amount of computing task is transferred to the target edge server. Transmission time from source to target edge server is expressed as follows:

$$T_i^{mig} = \frac{B_i(1-\theta_i-\beta_j)}{R_i} \tag{7}$$

where β_j is the amount of computing task performed by edge server j. The processing time of the task by target edge server j' is calculated as follows:

$$T_i^{exe'} = \frac{C_i B_i(1-\theta_i-\beta_j)}{f_i^{edge_{j'}}} \tag{8}$$

As a result, migration cost is included to the overall processing latency as follows.

$$T_i^{proc} = T_i^{trans} + T_i^{exe} + (T_i^{mig} + T_i^{exe'}) \tag{9}$$

E. Problem Formulation

Combining local computing, offloading and migration model, the system cost is defined as overall task processing latency, so the optimization function can be expressed in Eq. 10.

$$\min \sum_{i=1}^{M} T_i^{proc} \tag{10}$$

subject to

$$C1 : \theta_i \in [0,1], \quad \forall i \in M \tag{10a}$$

$$C2 : \sum_{i=1}^{M} f_i \le F_{\max} \tag{10b}$$

$$C3 : Z_i \le Z_{threshold} \tag{10c}$$

where $C1$ is the offloading variable decision between 0 and 1. $C2$ represents that the sum of computing resources allocated to the each user should be equal to or less than the total computing resources. $C3$ represents the acceptable latency to achieve the computing task.

IV. Proposed Reinforcement Algorithm

In this section, we present a Multi-Agent Reinforcement Learning (MARL) algorithm, which observes the state of the edge servers and end-users' requests in every time slot t, and determines the resource allocation strategy. In every time slot, new users' requests arrive and the resource is allocated depending on the optimization problem given in Eq. 10.

A. Multi-Agent Reinforcement Learning (MARL)

In the proposed DT-empowered MARL system, the agents exchange information with each other. Each agent gives decisions based on its local observation and environment with coordination [13]. The proposed MARL system consists of M cooperative agents, where each user is considered as an agent to learn the optimal resource allocation strategy and all agents collaborate with each other to achieve towards the same goal.

A direct generalization of Markov Decision Process (MDP) is Markov Games, also called as stochastic games. Stochastic games model dynamic interactions where the environment changes according to the choices of agents and therefore it is useful in modeling real-time situations [14]. In each stage, every agent in a given state selects an action from a set of actions to maximize joint utility. The actions which the agents choose define the stage payoff that each agent takes, as well as a probability distribution according to the new state that the agent will visit.

A stochastic game can be expressed by a tuple $(M, S, A, \mathbb{P}, R_i, \gamma)$, where M denotes the set of $(M > 1)$ agents, S and A represent the observed state space and action space, respectively. $\mathbb{P} : S \times A$ is the state transition probability function from any state $s \in S$ to $s' \in S$ for joint action $a \in A$. R_i determines the reward function for each agent $i \in M$. All agents collaborate with each other for a common reward function. $\gamma \in [0, 1)$ is the discount factor.

B. Joint Learning Strategy

Independent Learners (IL) implement Q-learning procedure in the traditional sense and they do not consider the existence of other agents. However, in our model, we consider a fully cooperative system where joint action learners (JAL) integrate both own action value and the action values of other agents. JAL is an agent that learns Q-values for joint actions [15]. Joint action means that each agent can observe the actions of other agents. Thereby, each agent will choose a strategy according to strategies of other agents. To achieve this, the Q function can be calculated by the following Bellman optimality equation.

$$Q_{\pi_i}(s,a) = \mathbb{E}\Big[r_t + \gamma \max_{a'} Q(s_{t+1}, a') | s_t = s, a_t = a\Big] \tag{11}$$

An agent aims to find best joint policy, $\pi : S \rightarrow A$ that could maximize the expected sum of rewards and defined as $\pi(a|s) := \prod_{i \in M} \pi^i(a^i|s)$. The value function of agent i, $V^i : S \rightarrow R$ for state s and any joint policy π at time t is as follows [13]:

$$V^i_{<\pi^i,\pi^{-i}>}(s) := \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t R^i(s_t, a_t, s_{t+1})|a^i_t \sim \pi^i(.|s_t), s_0 = s\Big] \quad (12)$$

It is important to note that $-i$ denotes all agents except agent i. In this model, we aim to find best performance of all agents. This can be provided by both own policy of the agent i and the decisions of all other agents since the capacity of edge servers is limited. Thus, $< \pi^i, \pi^{-i} >$ represents the joint strategy.

C. MARL Algorithm Design

Based on the joint learning strategy, we formulated our multi-agent resource allocation problem and main components in our MARL-based algorithm are determined as follows.

- *State:* The state $s \in S$ at time t is represented by a tuple consisting of the locations of the end-users, and the number of end-users connected to the each edge server.
- *Action:* Each end-user can locally or partially execute the computing tasks, offload to the edge server or migrate to the new edge server.
- *Reward:* After the agent takes an action a_t on state s_t, it obtains a reward as $r(s_t, a_t) = 1/T_i^{proc}$. Obviously, to maximize the reward, the overall processing latency should be minimized.

In this paper, JAL based multi-agent Q learning algorithm is proposed for multiple edge servers to jointly learn the best resource allocation strategy for the formulated problem. Algorithm 1 gives the details of the our DT-based solution.

Algorithm 1 JAL-based Resource Allocation Algorithm

```
1: Input: Constitute $N \times M$ bipartite graph with $N$ edge servers and $M$ end-users
2: for t = 1, ..., T do
3:     $Q(s, a^1, ..., a^M) \leftarrow 0$
4:     $V(s) \leftarrow 0$
5: end for
6: Observe the current state $s$
7: for i = 1, ..., M do
8:     Choose action $a^k$ for state $s$ using policy $\pi$
9:     Take action $a^k$
10:    Observe other agents' actions, $a^1, .., a^{k-1}, a^{k+1}, ..., a^M$
11:    Observe reward r and succeeding state $s'$ provided by environment
12:    $Q(s, a^1, ..., a^M) \leftarrow Q(s, a^1, ..., a^M) + \alpha[r + \gamma V(s') - Q(s, a^1, ..., a^M)]$
13:    $s \leftarrow s'$
14: end for
```

V. Performance Evaluation

In this section, we discuss the performance of our DT-empowered model. We compare our model with the local computing scheme and random assignment as benchmarks and investigate the impact of processing latency on the resource allocation. In the local computing scheme, the computing tasks are executed locally by end users while random scheme is that end users are uniformly allocated to edge servers or, computing tasks are executed locally or partially at random. We also evaluate two different strategies: (i) joint learning strategy (ii) independent learning strategy as described in Section IV-B. We vary the number of end users and the task size to measure the task processing latency. This is the time from task request time to task completion. We use Python and TensorFlow to evaluate the performance of our DT-empowered resource allocation algorithm.

TABLE I
Simulation parameters.

Parameters	Value
Edge server computation capability	10 GHz
End user computation capability	0.5 GHz
Bandwidth	$W = 10\ MHz$
End user transmission power	100 mW
Noise power	$\sigma^2 = 10^{-11}\ mW$
Channel power gain	$127 + 30\log(d)$
Delay requirement	1 s
Minimum data rate	0.1 $Mbps$

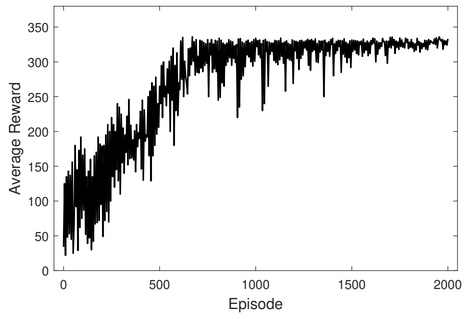

Fig. 4. Average reward with training episode of the proposed DT-empowered joint resource allocation algorithm.

A. Methodology

We consider a topology with $M = 15$ SBSs and varying number of end users in a $500m \times 500m$ area. It is assumed that each SBS is equipped with an edge server. End users connect to the nearest edge-server enabled SBS after authentication control and each edge server is able to serve the end users simultaneously. The number of newly arriving user requests in each time slot has a Poisson distribution with a rate of $(0, 0.2]$. The number of users per edge server ranges from 10 to 50. Simulation parameters are given in Table I.

First, we train our DT-empowered joint resource allocation strategy to show the effectiveness of our method. Fig. 4 shows the convergence of the average reward. In our model, maximizing the reward is provided by minimizing the system cost. System cost is formulated based on the task processing latency and migration time. The agent is with high probability of taking inappropriate action up to 1000 episodes. This results in low rewards and significant fluctuations. However, the reward function convergences after 1500 rounds of training and this demonstrates the effectiveness of our solution.

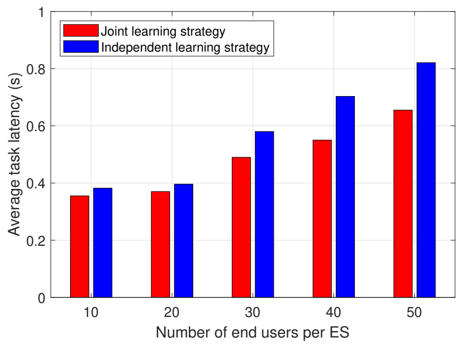

Fig. 5. Comparison of average task processing w.r.t the end users.

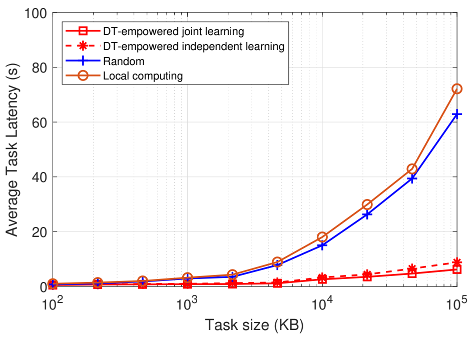

Fig. 6. Average task latency with different task size values.

B. Results

Fig. 5 shows the average task processing latency of joint learning strategy and independent learning strategy over the varying number of end users per edge server. In case of the joint learning, the agents cooperate with each other and learn optimal resource allocation strategy to minimize the latency. The digital copies of each agent can access both real-time and historical data, and observations of other agents connected to the same edge server, so they can cooperate with a subsets of agents in learning optimal policy. On the other hand, in case of the independent learning, the digital copies of each agent can only observe its local information. For ease of observation, we define the Eq. 10 as system cost. Local computing and offloading to the edge server are calculated by Eqs. 1 and 6, respectively. As the service migration decision is given, the task processing latency is calculated by Eq. 9. When the number of end users is increased, the average latency of joint learning strategy is always lower than independent learning strategy. We observe that the proposed model gains more importance in case of the high number of end users.

Fig. 6 compares the average task processing latency with different task size values. Larger task size causes greater cost to process the computation tasks. It is noteworthy that as the task size increases, the cost of random and local computing has a significant growth and end users need more computing resources. To meet the computation task demands, joint learning strategy gives the best solution with the interaction among agents.

VI. Conclusion

In this paper, we incorporate DT technology and an intelligent learning strategy to design an efficient resource allocation management in 6G-enabled massive IoT. In our model, digital twin helps allocate the computing resources among digital twin massive task demands with a cooperative MARL algorithm. Moreover, we implement an authentication control mechanism to prevent unauthorized access. We evaluate the performance of the our proposed approach in terms of task processing latency. The simulation results demonstrate that our approach has lower cost compared to the benchmark approaches.

Acknowledgement

Research supported by the German Federal Ministry of Education and Research (BMBF), 6G-RIC, grant 16KISK020K, 2021-2025.

References

[1] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, “Enabling massive iot toward 6g: A comprehensive survey,” *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 11 891–11 915, 2021.

[2] A. Masaracchia, V. Sharma, B. Canberk, O. A. Dobre, and T. Q. Duong, “Digital twin for 6g: Taxonomy, research challenges, and the road ahead,” *IEEE Open Journal of the Communications Society*, vol. 3, pp. 2137–2150, 2022.

[3] X. Liu, J. Yu, Z. Feng, and Y. Gao, “Multi-agent reinforcement learning for resource allocation in iot networks with edge computing,” *China Communications*, vol. 17, no. 9, pp. 220–236, 2020.

[4] N. Kiran, C. Pan, S. Wang, and C. Yin, “Joint resource allocation and computation offloading in mobile edge computing for sdn based wireless networks,” *Journal of Communications and Networks*, vol. 22, no. 1, pp. 1–11, 2020.

[5] J. Ahmed, M. A. Razzaque, M. M. Rahman, S. A. Alqahtani, and M. M. Hassan, “A stackelberg game-based dynamic resource allocation in edge federated 5g network,” *IEEE Access*, vol. 10, pp. 10 460–10 471, 2022.

[6] X. Xiong, K. Zheng, L. Lei, and L. Hou, “Resource allocation based on deep reinforcement learning in iot edge computing,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1133–1146, 2020.

[7] Z. Zhang, H. Zhouand, L. Zhao, and V. C. M. Leung, “Digital twin assisted computation offloading and service caching in mobile edge computing,” in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 1296–1297.

[8] D. Van Huynh, V.-D. Nguyen, S. R. Khosravirad, V. Sharma, O. A. Dobre, H. Shin, and T. Q. Duong, “Urllc edge networks with joint optimal user association, task offloading and resource allocation: A digital twin approach,” *IEEE Transactions on Communications*, vol. 70, no. 11, pp. 7669–7682, 2022.

[9] Q. Guo, F. Tang, and N. Kato, “Federated reinforcement learning-based resource allocation for d2d-aided digital twin edge networks in 6g industrial iot,” *IEEE Transactions on Industrial Informatics*, pp. 1–9, 2022.

[10] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, G. Karakostas, H. Wu, and X. Shen, “Digital twins from a networking perspective,” *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 525–23 544, 2022.

[11] X. Zhang, W. Wu, S. Liu, and J. Wang, “An efficient computation offloading and resource allocation algorithm in ris empowered mec,” *Computer Communications*, vol. 197, pp. 113–123, 2023.

[12] I. Labriji, F. Meneghello, D. Cecchinato, S. Sesia, E. Perraud, E. C. Strinati, and M. Rossi, “Mobility aware and dynamic migration of mec services for the internet of vehicles,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 570–584, 2021.

[13] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Cham: Springer International Publishing, 2021, pp. 321–384.

[14] E. Solan and N. Vieille, “Stochastic games,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 45, pp. 13 743–13 746, 2015.

[15] C. Claus and C. Boutilier, “The dynamics of reinforcement learning in cooperative multiagent systems,” ser. AAAI ’98/IAAI ’98. USA: American Association for Artificial Intelligence, 1998, p. 746–752.